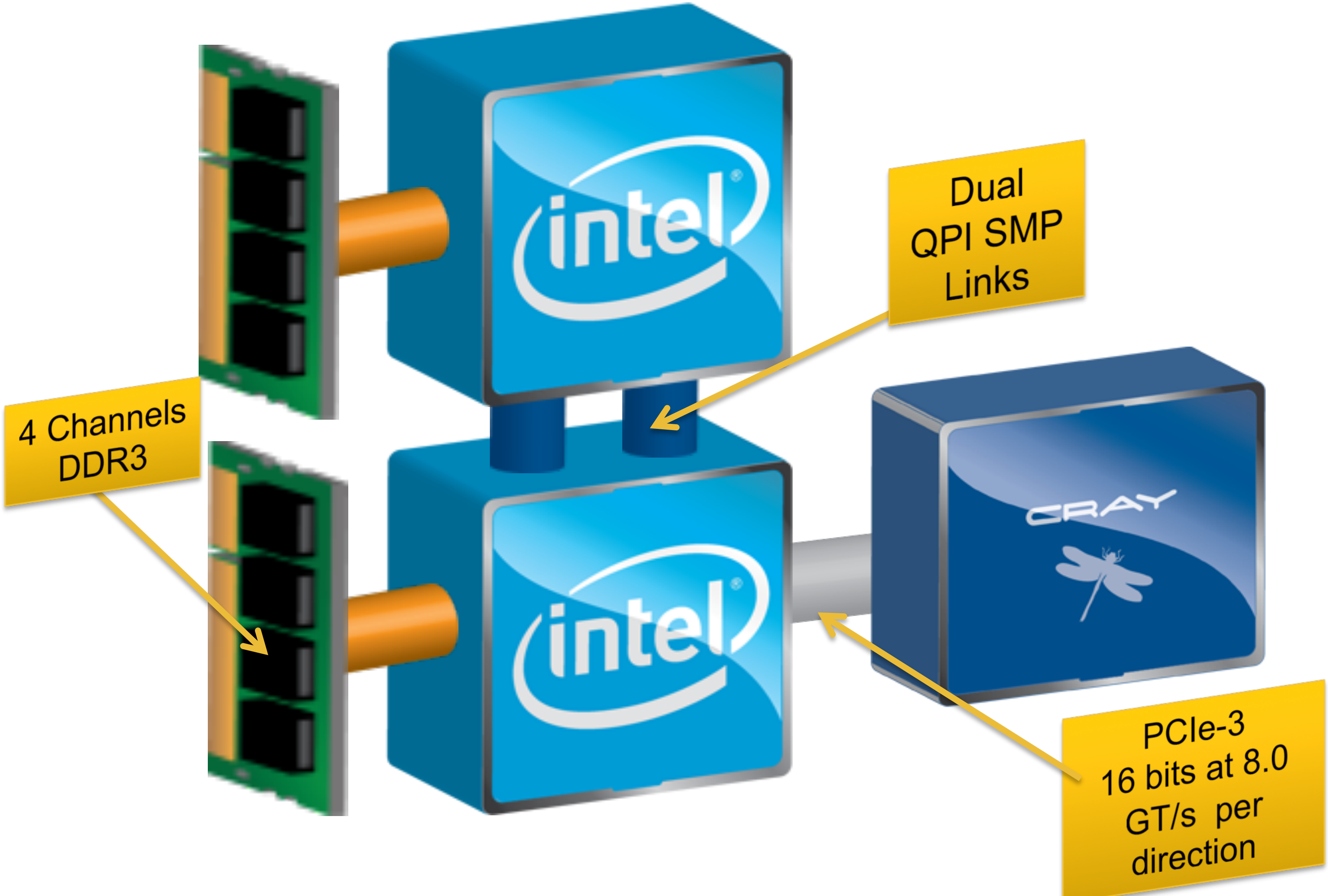
Cray XC30 System: Overview

Nathan Wichmann
wichmann@cray.com

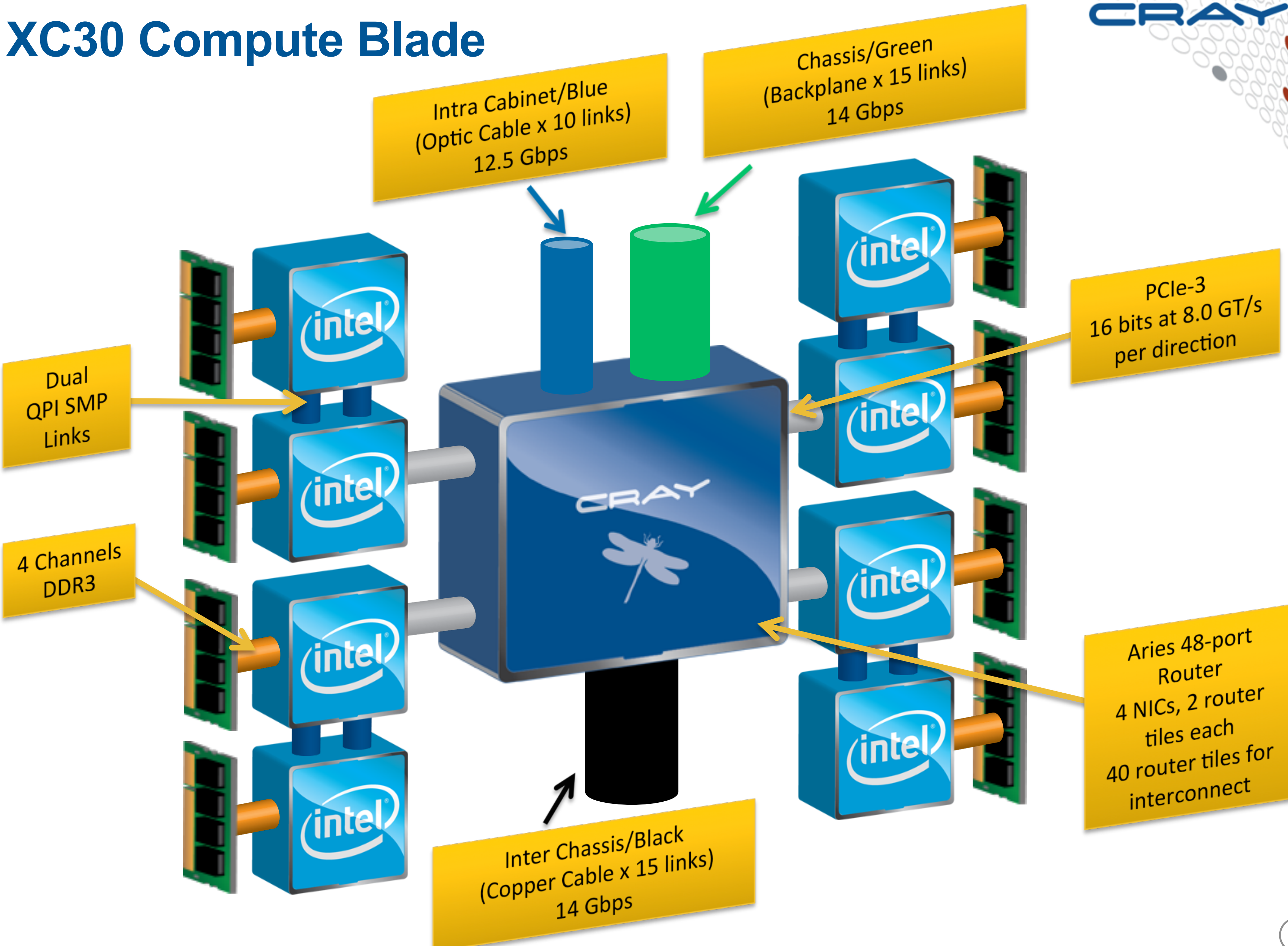
Outline

- Building Blocks
- A new compute node
- Dragonfly Topology
- Network and benchmark performance

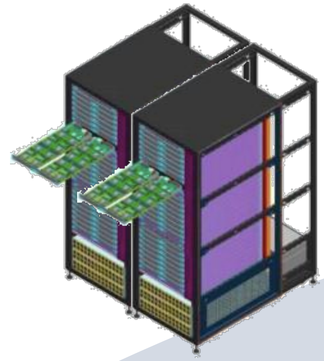
Cray XC30 Compute Blade Architecture

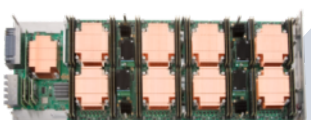


XC30 Compute Blade



Cray XC30 System Building Blocks





Compute Blade

4 Compute Nodes

Chassis

Rank 1 Network

16 Compute Blades

No Cables

64 Compute Nodes

Group

Rank 2 Network

Passive Electrical Network

2 Cabinets

6 Chassis

384 Compute Nodes

System

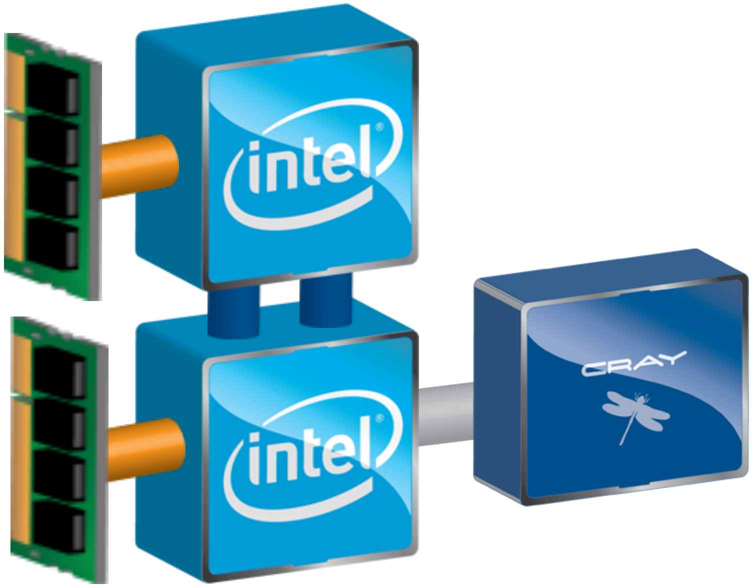
Rank 3 Network

Active Optical Network

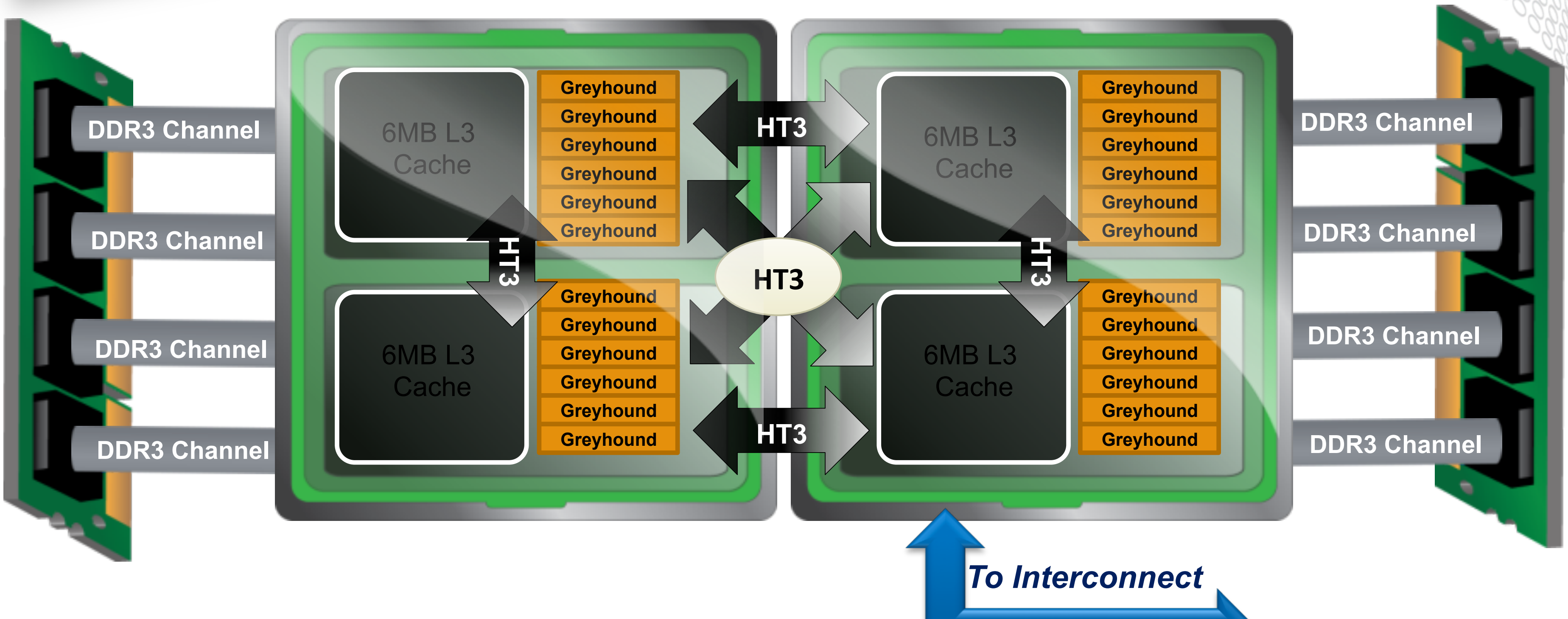
Hundreds of Cabinets

Up to 10s of thousands of nodes

Cray XC30 Compute node: Processor and environment comparison

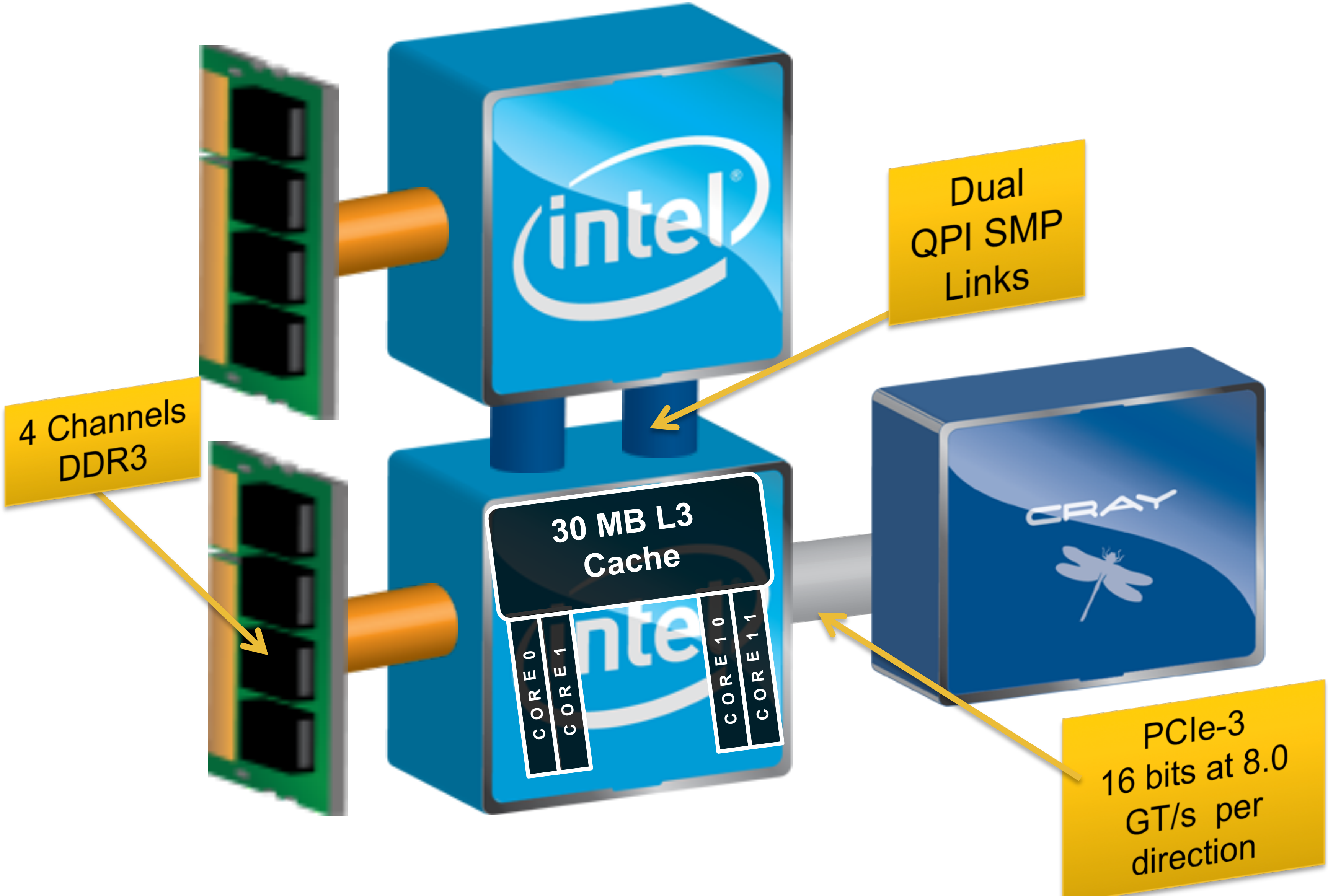


XE6 Compute Node Details: 24-core Magny Cours



- 2 Multi-Chip Modules, 4 Opteron Dies
- 24 (or 16) Computational Cores, 24 MB of L3 cache
- 8 Channels of DDR3 Bandwidth to 8 DIMMs
- Dies are fully connected with HT3

Cray XC30 Compute Blade Architecture



Magny Cours vs Ivybridge: bake-off

MAGNY COURS

- 6 cores per die
  - 4 die per node
- Each core has
  - 1 user thread
  - 1 SSE (vector) functional group
    - 128 bits wide
    - 1 add and 1 multiply
  - L1 cache size = 32 Kbytes
  - L2 cache size = .5 Mbytes
- L3 cache, size = 6 Mbytes
- Cache per core = .5 + 6/6 = 1.5 Mbytes
- Cache BW per core
  - L1 / L2 / L3 = 35 / 3.2 / 3.2 Gbytes/s
- Stream TRIAD BW/node = 52 Gbytes/s
- Peak DP FP per core = 4 flops/clk
- Peak DP FP per node = 96 flops/clk
- Memory latency = 110 ns

Ivybridge

- 12 cores per die
  - 2 die per node
- Each core has
  - 1 or 2 user threads
  - 1 AVX (vector) functional group
    - 256 bits wide
    - 1 add and 1 multiply
  - L1 cache size = 32 Kbytes
  - L2 cache size = 256 kbytes
- L3 cache, size = 30 Mbytes
- Cache per core= 30/8 = 2.5 Mbytes
- Cache BW per core
  - L1 / L2 / L3 = 100 / 40 / 23 Gbytes/s
- Stream TRIAD BW / Node = 100 Gbytes/s
- Peak DP FP per core = 8 flops/clk
- Peak DP FP per node = 480 Gflops
- Memory latency = 82 ns

Sandybridge and Ivybridge

Sandybridge

- 8 cores per die
  - 2 die per node
- Each core has
  - 1 or 2 user threads
  - 1 AVX (vector) functional group
    - 256 bits wide
    - 1 add and 1 multiply
  - L1 cache size = 32 Kbytes
  - L2 cache size = 256 kbytes
- L3 cache, size = 20 Mbytes
- Cache per core= 20/8 = 2.5 Mbytes
- Cache BW per core
  - L1 / L2 / L3 = 105 / 42 / 26 Gbytes/s
- Stream TRIAD BW / Node = 77 Gbytes/s
- Peak DP FP per core = 8 flops/clk
- Peak DP FP per node = 320 Gflops
- Memory latency = 82 ns

Ivybridge

- 12 cores per die
  - 2 die per node
- Each core has
  - 1 or 2 user threads
  - 1 AVX (vector) functional group
    - 256 bits wide
    - 1 add and 1 multiply
  - L1 cache size = 32 Kbytes
  - L2 cache size = 256 kbytes
- L3 cache, size = 30 Mbytes
- Cache per core= 30/8 = 2.5 Mbytes
- Cache BW per core
  - L1 / L2 / L3 = 100 / 40 / 23 Gbytes/s
- Stream TRIAD BW / Node = 100 Gbytes/s
- Peak DP FP per core = 8 flops/clk
- Peak DP FP per node = 480 Gflops
- Memory latency = 82 ns

Single Stream vs Dual Stream

- **Cray compute nodes booted with hyperthreads always ON**
- **User can choose to run with one or two ranks/pes/threads per core**
- **Choice made at runtime**

- **aprun –n### -j1 … -> Single Stream mode, one rank per core**
- **aprun –n### -j2 … -> Dual Stream mode, two ranks per core**

- **Default is Single Stream**
- **Dual Stream often better if…**
  - throughput is more important OR…
  - performance per node is more important OR…
  - your code scales extremely well
- **Single Stream often better if…**
  - single job performance matters more
  - per core performance matters most (code does not scale well)

- **Cray ended up running 4 or the 7 "NERSC SSP" codes in dual stream mode to maximize overall system score**

Core specialization

- **System 'noise' on compute nodes may significantly degrade scalability for some applications**
- **Core Specialization can mitigate this problem**
  - M core(s)/cpu(s) per node will be dedicated for system work (service core)
  - As many system interrupts as possible will be forced to execute on the service core
  - The application will not run on the service cpus
- **Use aprun -r to get core specialization**

  $ aprun –r[1-8] –n 100 a.out
- **Highest numbered cpus will be used**
  - Starts with cpu 31 on Sandybridge nodes
- **<u>Independent of aprun –j setting</u>**
- **apcount provided to compute total number of cores required**

  man apcount

Running with OpenMP and the Intel PE

- **An extra thread created by the Intel OpenMP runtime interacts with the CLE thread binding mechanism and causes poor performance**

- **To work around this issue cpu-binding should be turned off**
  - Allows user compute threads to spread out over available resources
  - Helper thread will no longer impact performance

- **Note: This is only an issue for running OpenMP programs that were compiled and linked with the Intel compiler**

Examples of using MPI and OpenMP with Intel PE

- **Running when "depth" divides evenly into the number of "cpus" on a socket**

  export OMP_NUM_THREADS="<=depth"
  aprun -n npes -d "depth" -cc numa_node a.out

- **Running when "depth" does not divide evenly into the number of "cpus" on a socket**

  export OMP_NUM_THREADS="<=depth"
  aprun -n npes -d "depth" -cc none a.out

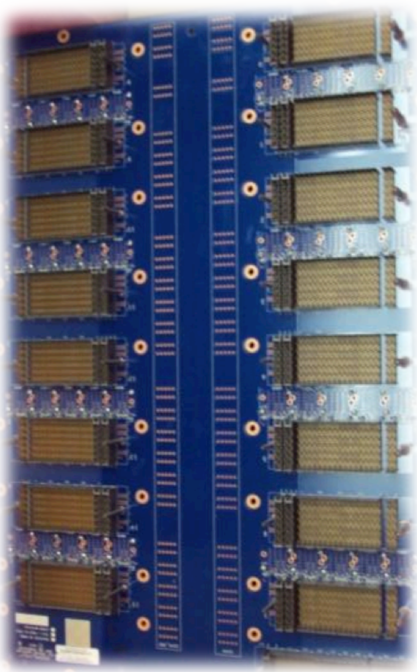
- **Take into account –j1 vs –j2**
- **These "-cc" options turn off cpu binding**
  - Your process/thread may switch cores in the middle of execution

- **Would LOVE to see a comparison of performance between shutting off binding and forcing binding**

Cray XC30 Dragonfly Topology

Cray XC30 Network

- The Cray XC30 system is built around the idea of optimizing interconnect bandwidth and associated cost at every level

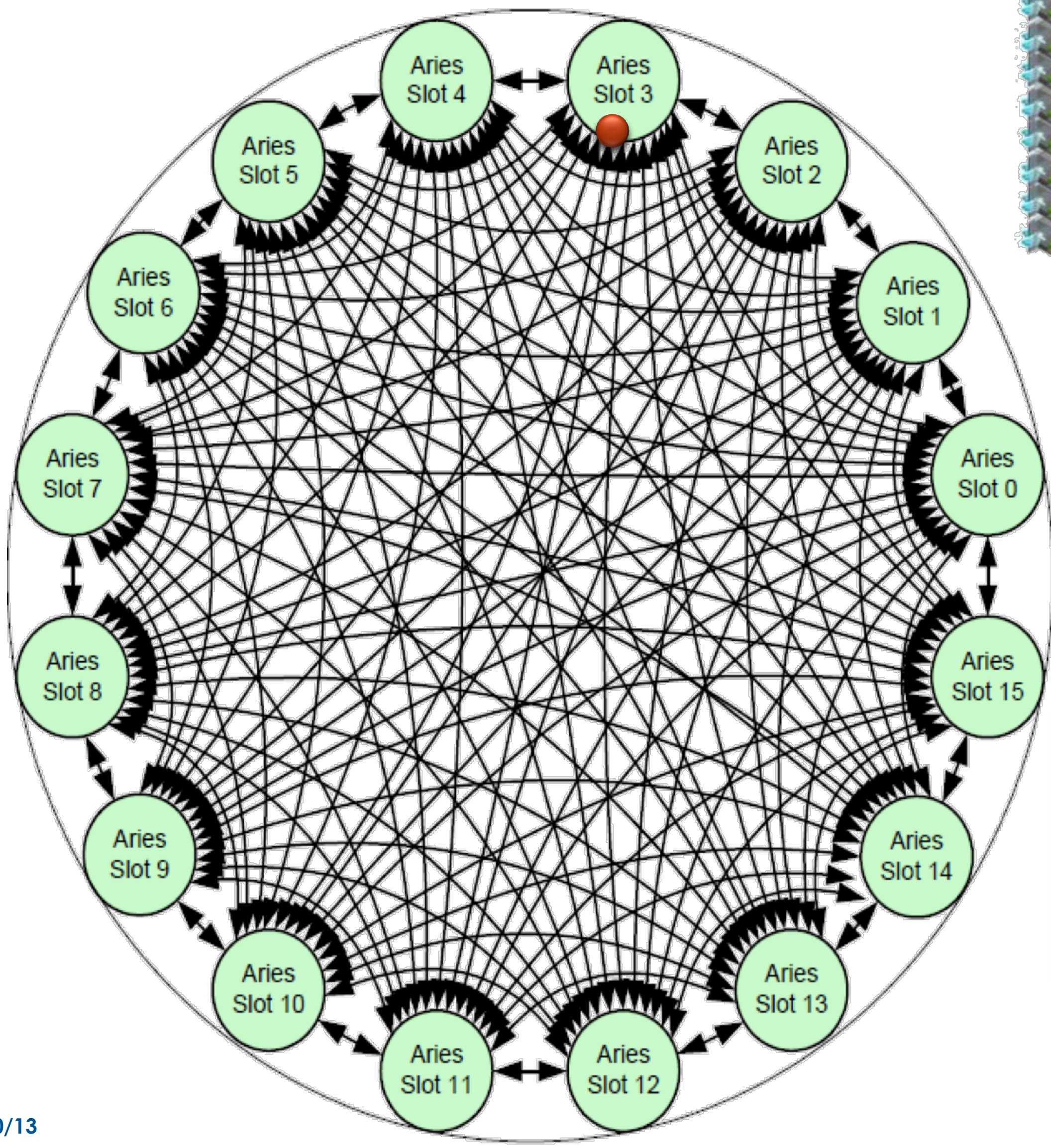


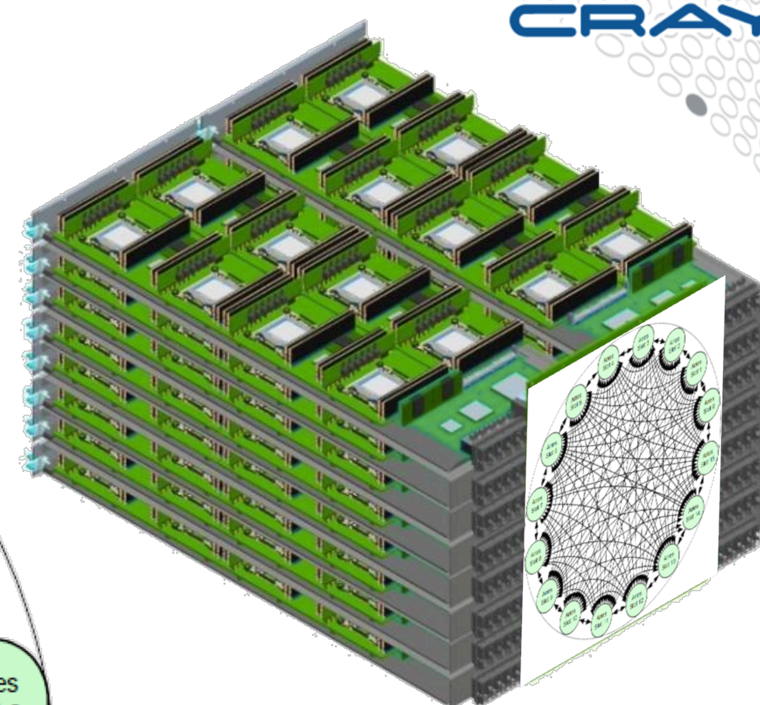
Rank-1
PC Board: ¢¢¢

Rank-2
Passive CU: $

Rank-3
Active Optics: $$$$

Cray XC30 Rank1 Network

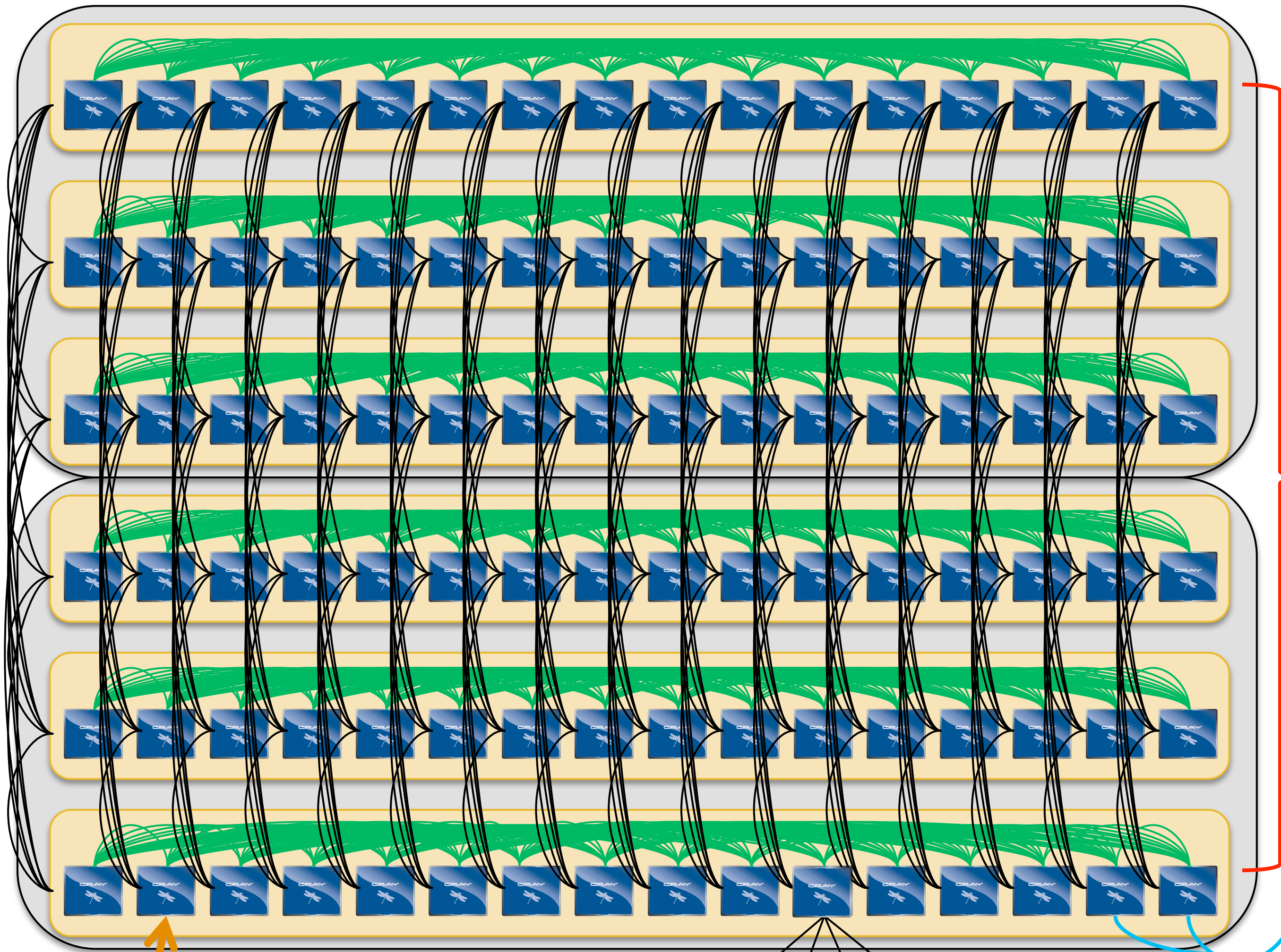


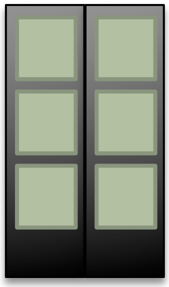


- Chassis with 16 compute blades
- 128 Sockets
- Inter-Aries communication over backplane
- Per-Packet adaptive Routing

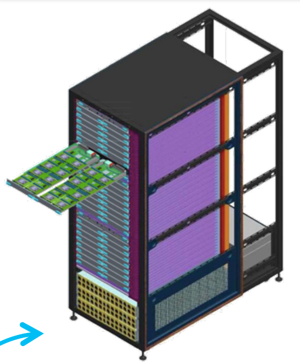
Cascade – Local Electrical Network

2 Cabinet Group
768 Sockets



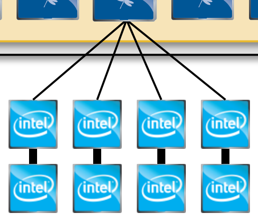


6 backplanes connected with copper cables in a 2-cabinet group: “Rank-2 Network”



Active optical cables interconnect groups
“Rank-3 Network”

16 Aries connected by backplane
“Green Network”

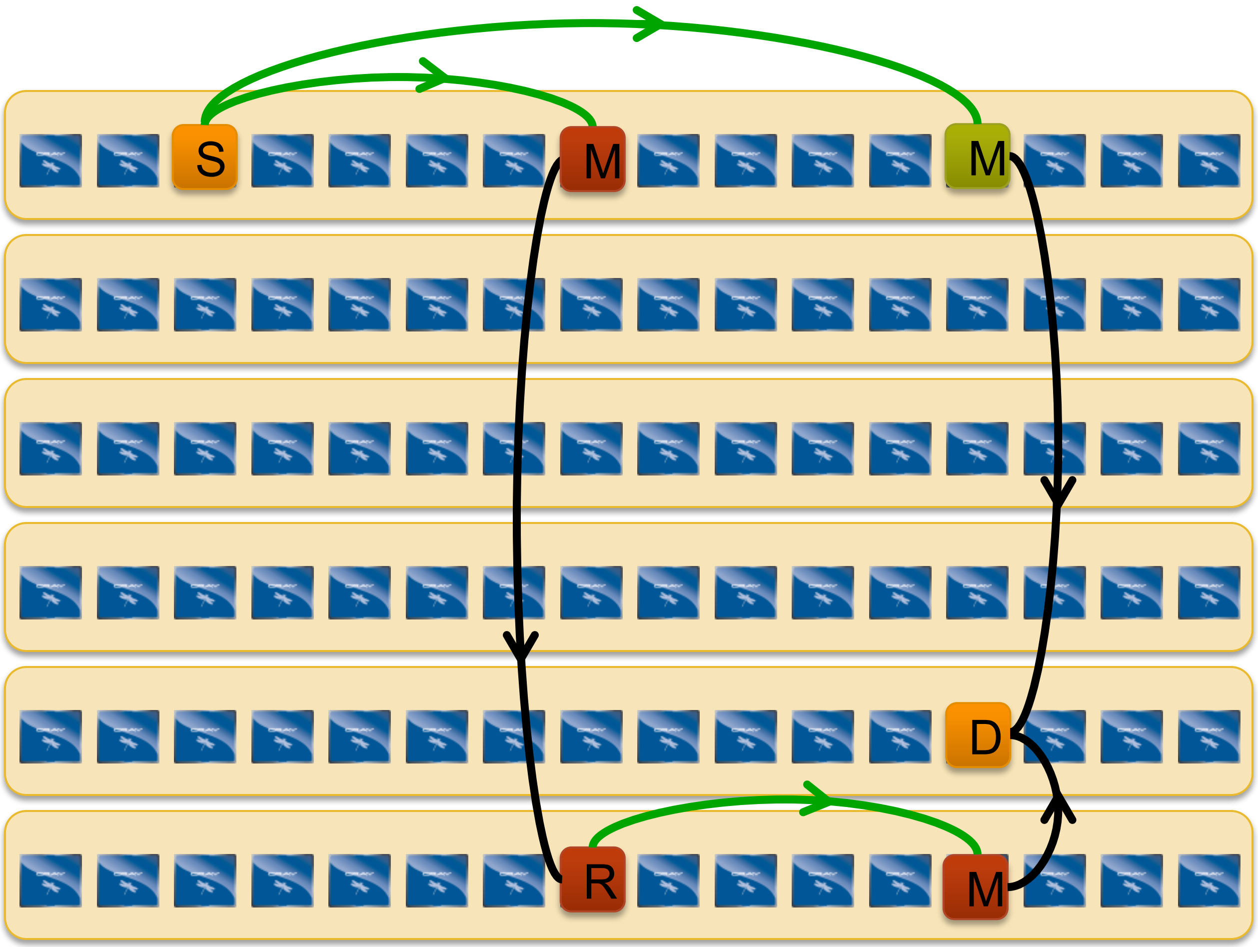


4 nodes connect to a single Aries

Cray XC30 Rank-2 Cabling

- Cray XC30 two-cabinet group
  - 768 Sockets
  - 96 Aries Chips

Cray XC30 Adaptive Routing



Minimal route between any two nodes in a group is just two hops

Non-minimal route requires up to four hops.

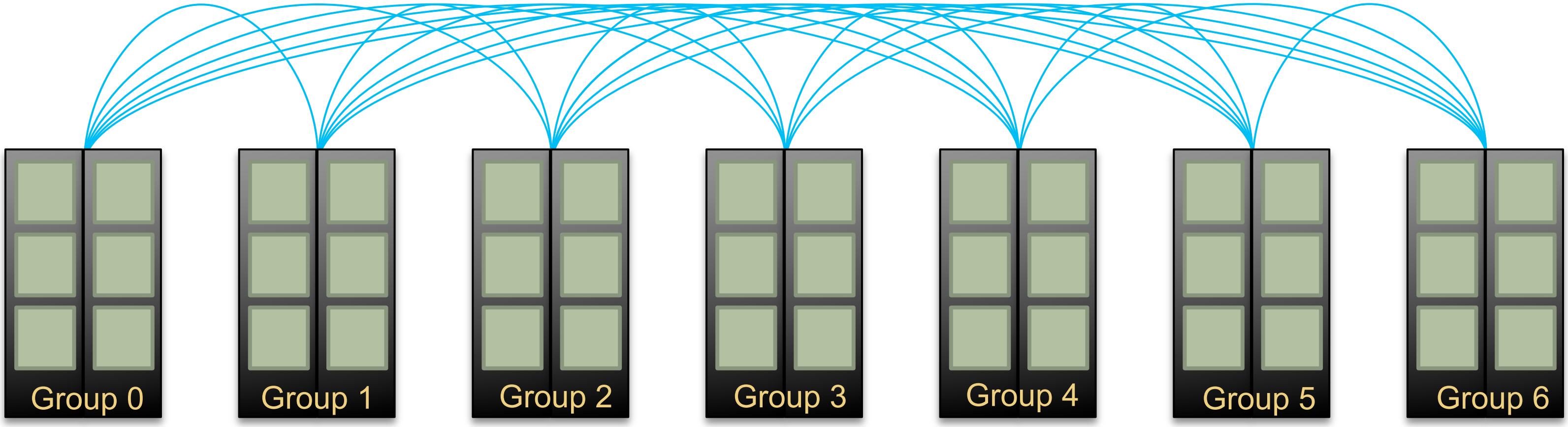
With adaptive routing we select between minimal and non-minimal paths based on load

The Cray XC30 Class-2 Group has sufficient bandwidth to support full injection rate for all 384 nodes with non-minimal routing

- **Adaptive routing allows the Cray XC network to handle a diverse set of traffic patterns at full speed**
  - Significant advantage over Infiniband on real traffic patterns

Cray XC30 – Rank-3 Network

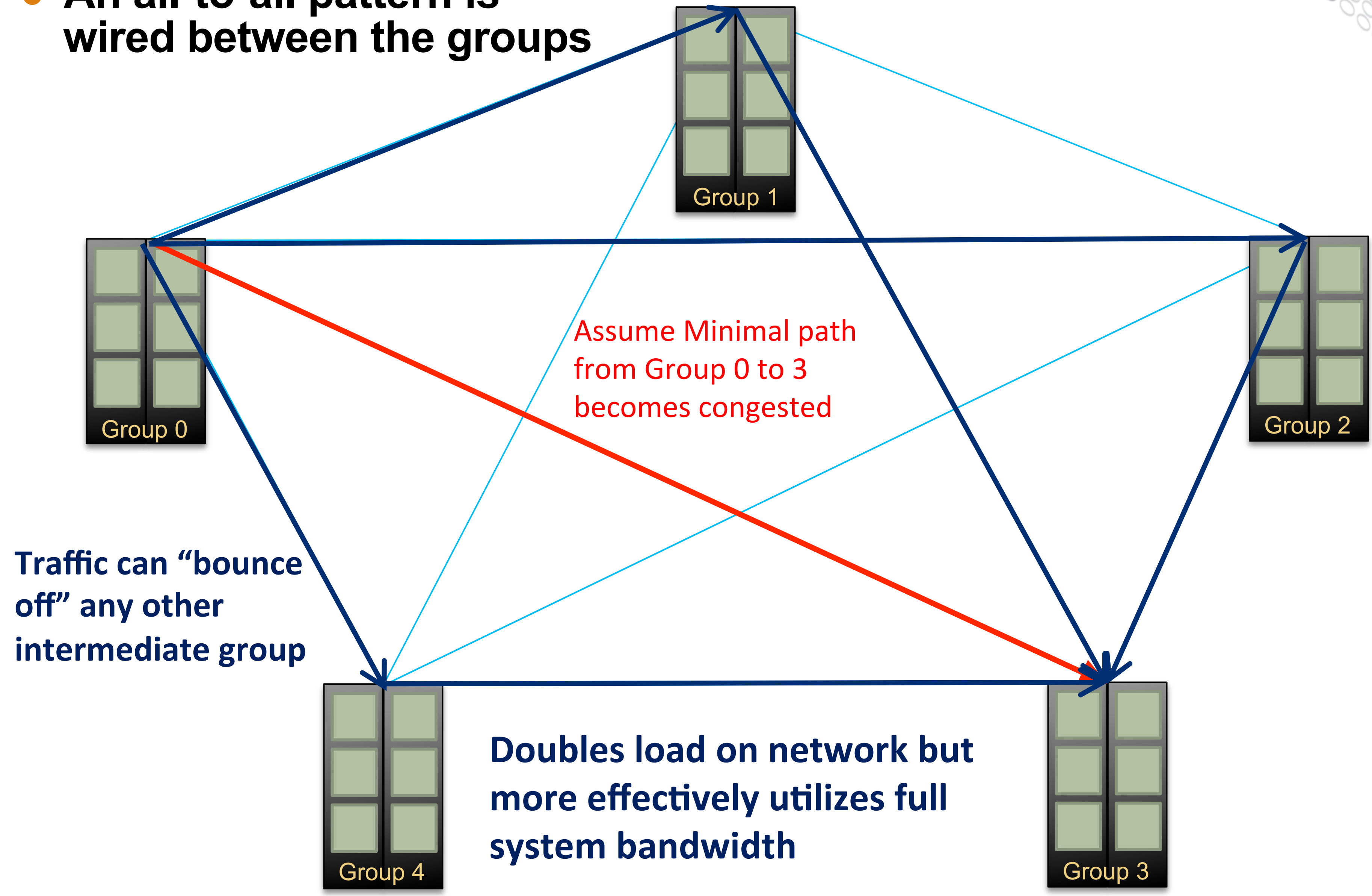
- An all-to-all pattern is wired between the groups using optical cables (blue network)
- The global bandwidth can be tuned by varying the number of optical cables in the group-to-group connections



Example: A 7-group system is interconnected with 21 optical "bundles". The "bundles" can be configured between 2 or more cables wide, subject to the group limit.

Adaptive Routing over the Blue Network

- An all-to-all pattern is wired between the groups



Why use Huge Pages?

- **On edison huge pages are a performance enhancement**
  - On hopper hugepages were a functional requirement for some codes
- **The Aries may perform better with HUGE pages than with 4K pages.**
  - HUGE pages use less Aries resources than 4k pages
  - More important when remotely access large percentage of nodes memory in an irregular manner
    - Large AlltoAll
    - AMO GUPS
- **Still be watchful for memory page fragmentation**
  - Might still get "cannot run errors" because it cannot find enough large hugepages
- **Use modules to change default page sizes (man intro_hugepages):**
  - e.g. module load craype-hugepages#
    - craype-hugepages2M
    - craype-hugepages8M
    - craype-hugepages16M
    - craype-hugepages32M

MPI Latency and Bandwidth

Multipong Benchmarks		
Test Description	**Measured**	**Units**
Maximum Inter-Node Latency Single-Core, Farthest-node pair (1)	1.920	µsecs
Minimum Inter-Node Latency Single-Core, Nearest-node pair (2)	1.498	µsecs
Maximum Intra-Node Latency Single-Core, cross socket (3)	0.545	µsecs
Minimum Intra-Node Latency Single-Core, same socket (4)	0.267	µsecs
Maximum Inter-Node Latency Fully-packed Nodes, Farthest-node pair (5)	2.452	µsecs
Maximum Inter-Node Latency Fully-packed Nodes, Nearest-node pair (6)	2.027	µsecs
Maximum Bandwidth Multi-Core Nearest Nodes (7)	9255	MB/s

Point-to-Point Aries vs. Gemini

Typical Point-to-point bandwidth		
Case	Gemini	Aries
	(GB/s)	(GB/s)
"On Gemini/Aries"	~5	~8-10
"Long Range"	~1.5-3	~8-10

- Long Range transfers on Aries will be able to adapt around any hot spots in the network and continue at full speed
- Maximum latency will be much lower on Aries

Optical network rarely limiting factor in real life

- **Most traffic patterns will be limited by the sustained injection bandwidth**
  - Sustained injection bandwidth is in the 3-6 Gbytes/sec range
  - Nearest Neighbor communication mostly stays on-group
- **Examples of optical bound benchmarks**
  - Full system alltoall with <50% of optical cables connected
  - Pure bi-section bandwidth tests, but that is not common in real codes
  - Global bandwidth intensive codes that are packed into just a few groups
    - Seems unlikely to occur in production
- **Less then full system runs are unlikely to be optical limited**

- **Communication intensive applications are more likely to be injection bandwidth bound rather than network bound**
  - Consider optimization that maximize on-node traffic and minimize off-node traffic

Additional Network test

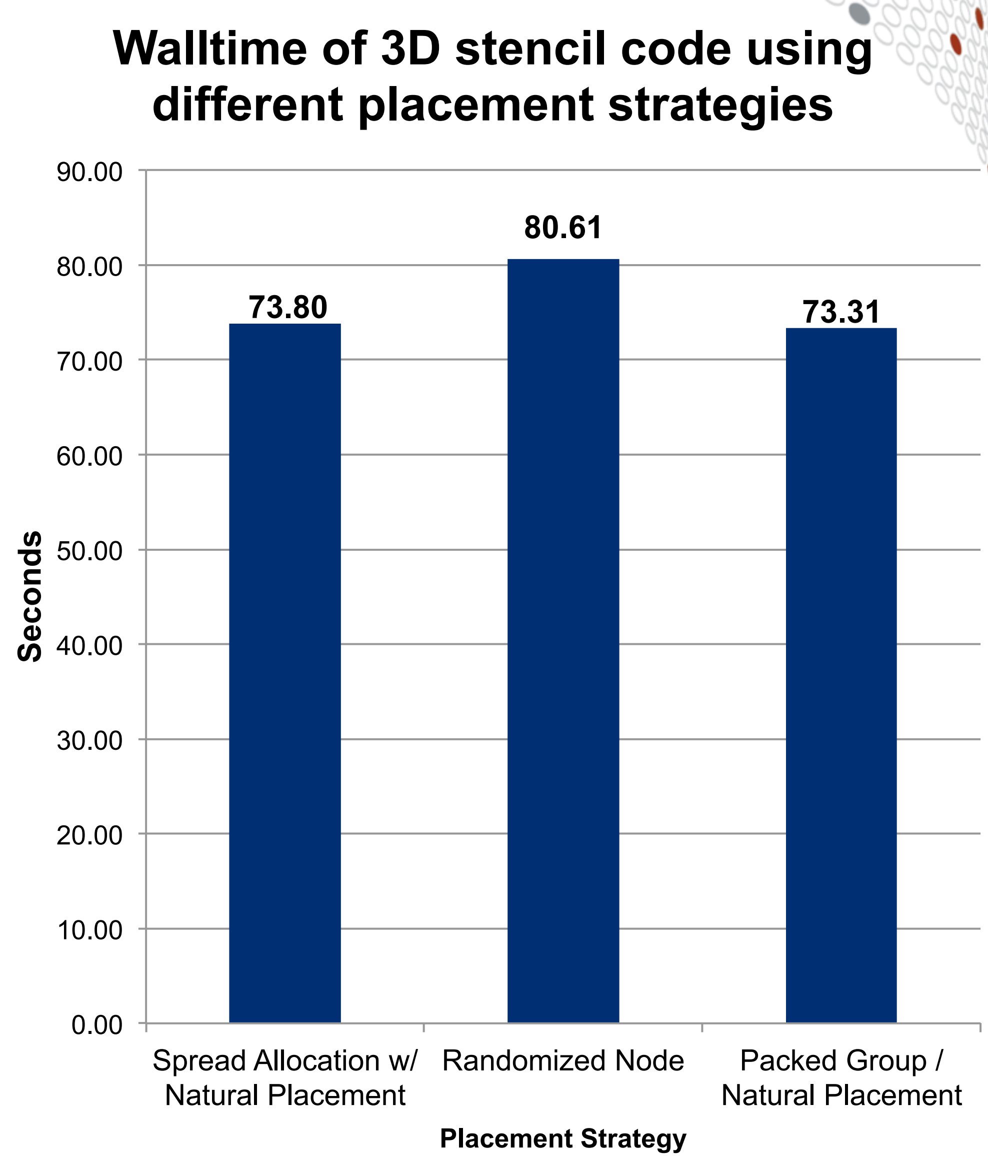
- MPI_ALLTOALL
  - Dmapp optimization during communication in available under 6.0
    - MPICH_USE_DMAPP_COLL = 1
  - Measured ~ 9 Tbytes / second of global bandwidth
    - Very good performance for this configuration

- MPI Barrier / Allreduce – excellent scaling with dmapp version

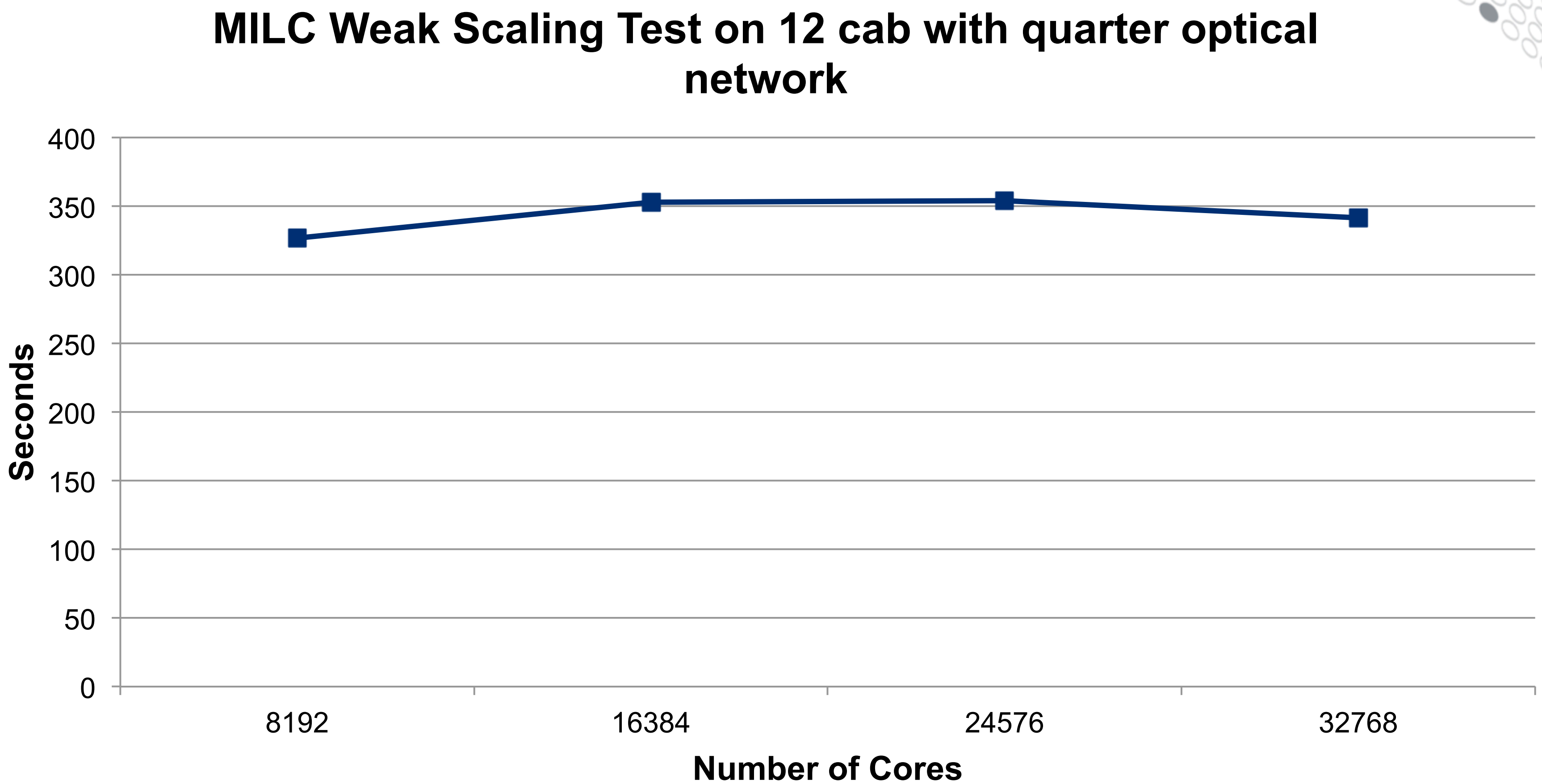
- Initial conclusions:
  - High speed network is healthy and performing well.
  - Full system performance is very good.
  - Adaptive routing working very well (as designed).

Placement Strategies impact on 3D stencil

- Run on 14 copies of a 3D stencil code simultaneously on a 28 cabinet system with partial optical network
- Spread Allocation w/ Natural Placement
  - Spread across the machine
  - "Naturally" fill your portion of the group before moving on to the next group
  - Preserves some spatial locality while still spreading out the job
- Randomized Node
  - Spread across the machine
  - No spatial locality
- "Packed Group" fills a cabinet before moving on to the next cabinet
  - Maximizes on-group traffic
- Conclusions
  - Natural placement a good idea
  - Don't destroy spatial locality
  - Pack Group slightly better, but performance is not hurt significantly if job gets spread out



Near perfect scaling of MILC



- **MILC does a 4D Nearest Neighbor Halo exchange**
  - Cause significant network contention on a 3D torus
  - Significant amounts of traffic stays on group
  - Also sets up patterns were all off-group traffic goes to one other group
  - Would only work well if adaptive routing was working well

Summary

- **On-node**
  - 24 cores per node on edison; similar to hopper
    - Edison has new –j1 vs –j2 (hyperthreading) feature
  - Edison has ~2X the bandwidth of hopper per node
  - Intel compiler now available
- **Network**
  - Edison has improve injection bandwidth over hopper
  - Edison has a greatly improved network bandwidth
    - Global bandwidth is significantly higher
    - Adaptive routing minimizes hot spots
    - Better scaling
    - Less job-job interference
  - Communication intensive applications more likely to be injection bandwidth bound rather than network bound
- **Overall application performance should be significantly improved compared to hopper**

The End